SMU Data Science Review

Volume 1 | Number 2 Article 13

2018

Stack Overflow Question Retrieval System

Vishi Cline
Southern Methodist University, vcline@mail.smu.edu

Abhishek Dharwadkar
Southern Methodist University, avdharwadkar@mail.smu.edu

Rajni Goyal
Southern Methodist University, rajnig@smu.edu

Daniel Engels
Southern Methodist University, dwe@smu.edu

Raghuram Srinivas
Southern Methodist University, rsrinivas@mail.smu.edu

See next page for additional authors

Stack Overflow Question Retrieval System

Authors

Vishi Cline, Abhishek Dharwadkar, Rajni Goyal, Daniel Engels, Raghuram Srinivas, and Sohail Rafiqi

Stack Overflow Question Retrieval System

Abhishek Dharwadkar[1], Vishi Cline[1], Rajni Goyal[1]
Daniel Engels[2], Raghuram Srinivas[2], Sohail Rafiqi[2]

[1] Master of Science in Data Science, Southern Methodist University
6425 Boaz Lane, Dallas, TX 75205
{avdharwadkar, rajnig, vcline, rsrinivas}@mail.smu.edu
{dwe, srafiqi}@lyle.smu.edu

Abstract. In this paper, various approaches were presented to match the most similar question to a user's query. This is a two-step process, wherein the tags/topics of the questions are identified using k-means clustering and topic modeling respectively. User's query is then matched with the most similar question in the corpus using k-means, topic modeling and ensemble models. Our motivation is to improve the developer's productivity by presenting the top 10 most relevant questions similar to the users' query. Our study is focused on answering Python (windows) specific technical programming related questions using the Stack Overflow dataset. The models are built using k-mean classification, topic modelling and ensemble of the two approaches, to find similar questions. These three approaches were chosen because the tags provided by the dataset were too generic to contextualize the question – which may result in irrelevant answer queries for future questions. Recall is the metric used to evaluate the models. Based on the results, we concluded that NMF and ensemble method outperformed k-means, with recall for NMF and Ensemble being 67% and recall for k-means being 50%.

1 Introduction

Stack Overflow contains plethora of information related to wide range of topics in computer programming. The user posting the question must specify a category of the question by selecting specific tags that the user thinks are most relevant to the context of the question. The user may choose the tag from a list of already existing tags that other users have already defined, or they may decide to define a completely new tag. It is recommended that the user always favor existing tags and only create a new one when they feel like there is strong evidence that the question they are posting covers a new topic that nobody else has asked about before at this site. This allows for quick retrieval of answers for future questions. However, the question poster typically specifies several tags and currently, there is no way of prioritizing those tags. In addition, the tags available in Stack Overflow may not be suitable or may be too generic to contextualize the question – which may result in irrelevant answer queries for future questions. Since the users assigns the tag, they are also subjective and open to user's interpretation. Several publications have been done using stack over flow data exploring wide variety of topics. Some of these

topics, which are relevant to our current research include but are not limited to: finding expert users in community question answering, trying to find deficient project documentation using topic analysis, answering questions about unanswered stack overflow questions, predicting tags for Stack Overflow posts and several others, which can be found in the citations.

While finding relevant answers and improving the quality of the tags has been an ongoing effort, the excessive number of answers and the click rate to reach a valid answer is another challenge. Stack Overflow is a question-answer search engine which is not personable and may result in delayed access to information.

The objective of our study is to present the user with top 10 most relevant questions like the users' query. We achieve this by developing a k-means classification model which labels the question with the most relevant tag or context. Thereafter we use topic modeling and ensemble techniques to find similarities between the users' query and questions in the corpus. We randomly chose 10 questions specific to python (windows) to evaluate our model and measured the accuracy of identified questions. The scope of our project is limited to a sample of questions related to python and windows. Within the python domain, only questions related to windows are included. Based on the user's question, we identify the intent and present the user with the top 10 most relevant questions. We measure the goodness of our question retrieval system by using recall rather than accuracy. For our study, recall is defined as the number of correct results divided by the number of results that should have been returned. We measure the relevancy of the similar questions through user feedback.

2 Stack Overflow

Stack Overflow is a common venue for developers to participate and share their programming knowledge using natural language text. This is a privately held website created by Jeff Atwood and Joel Spolsky in 2008. This website allows the users to ask questions and provide answers. The user can also vote the questions and answers up or down and upon successfully solving the question, mark it as the best answer. The best answer is not always the correct answer but may be specified as the correct answer based on helpfulness factor. Once the question is marked closed, no more answers can be posted. Every time the user makes a valued contribution on the site, the user earns reputation points and badges, which gives them access to elevated privileges. The user also has the ability to assign tags for their questions. Tags are chosen from a predefined list or the user can specify a new tag if there are no relevant tags available. The tags assigned by users are highly subjective and may or may not correctly contextualize the question.

In a publication authored by Clayton Stanley[7] and Michael D. Byrne[7], the authors built a system that automatically predicts the tag of a question on the stack overflow system. This publication explores three different methods to automatically assign the tags to a question. The first method uses a programming language detection-based approach which

trains the data using multinomial Naïve Bayes classifier. In the second method, they implement a SVM based classifier. While in the third method, they combine the two to create a hybrid system. Their model can successfully tag 65% of Stack Overflow posts correctly, when tasked to predict one tag per post on average. Another publication, conducted by members in faculty of computer science at Dalhousie University[6] in 2012, investigated a method to improve the accuracy of the answer by matching the question with the experts in that domain. They model the interests of users by tracking their answering history in the community. They use two different statistical topic models for solving this issue and compare these methods against the traditional information retrieval approaches, namely, language models with Dirichlet smoothing, TF-IDF, the Latent Dirichlet Allocation and Segmented Topic Model. They also show that Segmented Topic Model out-performs Latent Dirichlet allocation model. While these publications focused more on leveraging domain specific lexical rules based natural language processing techniques, Chunyang Chen[5], Zhenchang Xing[5], and Ximing Wang[5] published a paper in 2017, which discusses the use of natural language processing to build a thesaurus of software-specific terms and commonly used morphological forms to normalize software engineering text. They combine distributed word semantics, domain specific lexical rules and transformations, and graph analysis of morphological relations and measure the accuracy of their resulting thesaurus against the community-curated lists of software specific terms, abbreviations, and synonyms. They furthermore compare their thesaurus against the general NLTP pre-processing techniques to show that their thesaurus driven from software specific terms and their abbreviations/synonyms perform better for text normalization than the general NLTP methods.

These publications, while all in one way or the other related to our study, have different end goals, but can be used as ground work for our study. Each of these publications take a different approach; however, the end goal ultimately is to improve the accuracy of the answers and relevancy of the tags. Stack Overflow is built to provide answers to technical questions, but the accuracy of that answer is dictated by a single user who initially asked the question. Furthermore, the tags assigned for the question are highly subjective, may not be accurate, or may be too general to define the question. The potential inaccuracy of the marked answer and the assigned tags can drive down the recall and precision of the presented answers. Since Stack Overflow is set up as a question/answer engine, it doesn't allow the user to cross reference all the questions or threads that are related to the question they are asking. Hence, it is not necessary that the wisdom of the crowd is getting leveraged. We have built a classification model to identify a more relevant tag for the question. We measured the accuracy of the tag by conducting a user study. In addition, based on the users' question, we identify the intent and build models to find similar questions. Recall is the metric used to measure the similarity of the question.

3 Data

We initially attempted to download the Stack Overflow dataset from the archive. However, the archive dataset was 10 Gigabytes zipped. Due to our local computer configuration restrictions and inability to work with such a large dataset, we opted to use the dataset provided by Kaggle. Kaggle contains a dataset specific to python related questions. This dataset contains questions up to 19 October 2016 (UTC). Due to space limitations, they have only included non-deleted and non-closed content. The dataset contains 3 files: questions with 607282 entries, answers with 987122 entries, and tags with 1885078 entries. The major attributes of these tables are described below in Table 1, Table 2 and Table 3.

Table 1: Question Attributes

Column Name	Column Description
Id	Unique identifier for the question
Title	The title a question is asked with
Body	The raw body question text a post is submitted with.
Creation Date	The date that the question was created
Score	Total number of upvotes minus the downvotes based on the quality/helpfulness of question, as rated by the open community of users. This value may be a negative value.
Owner ID	Identifier of the user posting the question

Table 2: Answer Attributes

Column Name	Column Description
Id	Unique identifier for the answer
ParentID	Foreign key to the ID column in the Question table
Body	The raw body answer text a post is submitted with.
Creation Date	The date that the answer was created
Score	Total number of upvotes minus the downvotes based on the quality/helpfulness of the answer, as rated by the open community of users. This value may be a negative value.
Owner ID	Identifier of the user posting the answer

Table 3: Tag Attributes

Column Name	Column Description
Id	Foreign key to the ID column in the Question table
Tag	A word or a phrase that describes the topic of the question. Maximum of 5 tags can be created.

The attributes that we focused in our study are particularly the Question Body, the Answer Score, and the Tags assigned to each question. We filtered the dataset with 'python' and 'windows' tags, which resulted in a total of 5135 questions. The Question body is preprocessed, tokenized and used for feature extraction.

3.1 Exploratory Data Analysis

Exploratory Data Analysis is an important step in the discovery process. Once we chose our dataset, we need to visualize the data, explored the features we wanted to plot, and determined how to explore the data without having any preconceived bias. Exploratory Data Analysis helped us in answering important questions and found data anomalies and constraints which might have negatively or positively impacted our model.

Based on our analysis of the question and answer file, we discovered that there are 6,212 questions and 5,367 answers that have null owner id values. Also, there are 313 questions that have an empty tag associated with it. Since, in this paper we are not concerned with the user data, we discard empty owner ids. However, we might have to explore the empty tags further depending on if the tags are relevant or not in our analysis which we will be exploring in this paper. about empty owner ids. However, we might have to explore the empty tags further depending on if the tags are relevant or not in our analysis which we will be exploring in this paper.

We also looked at the character lengths of the question title, question body, and answer body. The question title ranges between a minimum of 10 characters and maximum of 172 characters with an average question title character length of 51 characters per question. As for the question body character length, it has a minimum length of 22 characters, a maximum of 48,242 characters, and an average of 1357 characters per question. As for the answer body, the minimum character length is 18 characters, the maximum is 33,888 characters and an average of 774 characters per answer.

Next, we looked at when are the python questions and answers being posted in terms of day of the week. In Fig. 1 and Fig. 2, we saw that on an average most questions and answers are being posted on a Wednesday and it tapers down towards the end of the week. The questions and answers posted are relatively lower on a weekend than a weekday. We also looked at the time to get a question answered for all of the questions and found that the average time a question will get answered is 36 days.

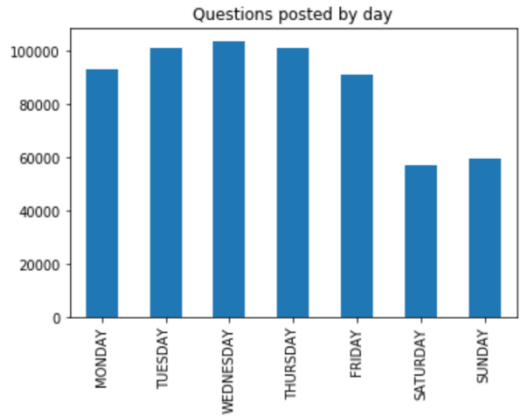

Fig. 1. Question counts by day of the week

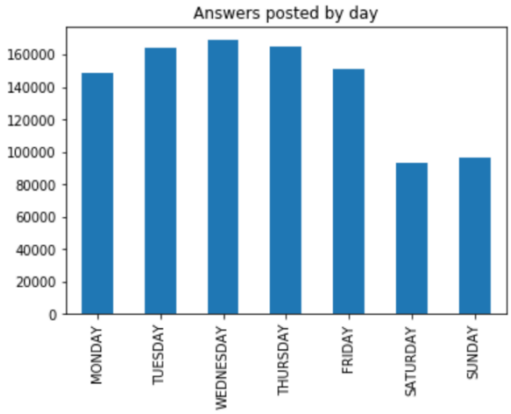

Fig. 2. Answer counts by day of the week

In the question and answer file, the score attribute has negative values in some records. After researching we found that the score of a question and answer is a computed field. It is calculated by “number of up votes - number of down votes”. Negative score implies that down votes exceed the up votes. Voting up is done by users when they find a question, answer or comment useful and appropriate. Voting down, also known as "casting down votes", is done by users whenever they encounter a question or an answer that is not deemed useful, clear or correct. Hence, a negative value for score attribute is a valid score. With

regards to the scores, there is a high correlation (0.66) between the average questions score and average answers score as shown in Fig. 3.

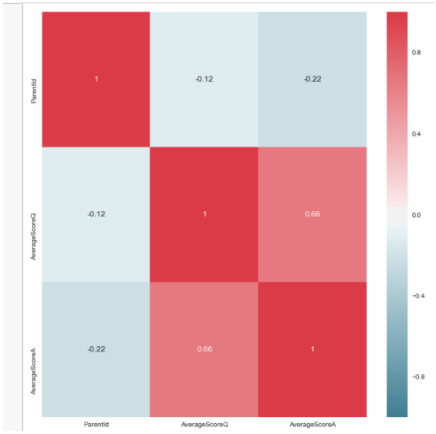

Fig. 3. Correlation between question and answer scores

Finally, we explored the tags dataset. Since the dataset was extracted for all the questions which had python as one of the tags, we excluded the tag "python" from the tags table for analysis. We observed that almost 10% of the python questions were related to Django (62,818), followed by pandas (26,852), numpy (25,848), list (18,951), matplotlib (16,521), and so on. There are tags that specify if the question applied specifically to python 2.7 (34,616) or python 3.x (26,814) versions.

3.2 Data Pre-Processing

Text pre-processing is one of the most important concepts prior to applying NLP and text analytic techniques, as components extracted from the text, whether it's tokenized sentences, words or phrases, are the foundation of future analysis. Cleaning and standardizing the text, as well as extracting more information regarding the text from the annotations are key to improving the accuracy of the classifiers.

First, we cleaned the text by removing html tags, removing stop words and other un-necessary terms, and conducting case conversions. Next, we used the NLTK framework to perform word tokenization, specifically the regexp tokenizer class, which tokenizes based on regular expression-based patterns to segment sentences and then segmented those sentences into words. We normalized the tokens to get clean and standardized version of the tokens. The techniques used for cleaning the tokens included conducting spelling corrections and performing stemming and lemmatization. We also expanded the

contractions, since contractions are just different representations of the same word, using a contraction mapping dictionary.

3.3 Model Pipeline

After data has been pre-processed and normalized, we performed text classification or categorization. In our problem context, text classification is trying to organize the text into categories based on the attributes of each text in the questions. In order to accomplish this, feature extraction and supervised/unsupervised machine learning algorithms are used. We created unsupervised classification model and assumed that we do not have any pre-labeled data and used k-means document clustering to cluster the questions together based on their inherent properties and assign labels to them. We used topic modeling to do content-based classification, which is a type of text classification where priorities/weights are assigned to topics in the text, which helps in determining the category of that class. In addition, we also did an ensemble of k-means and Topic Modeling. The model pipeline is displayed in Fig. 4.

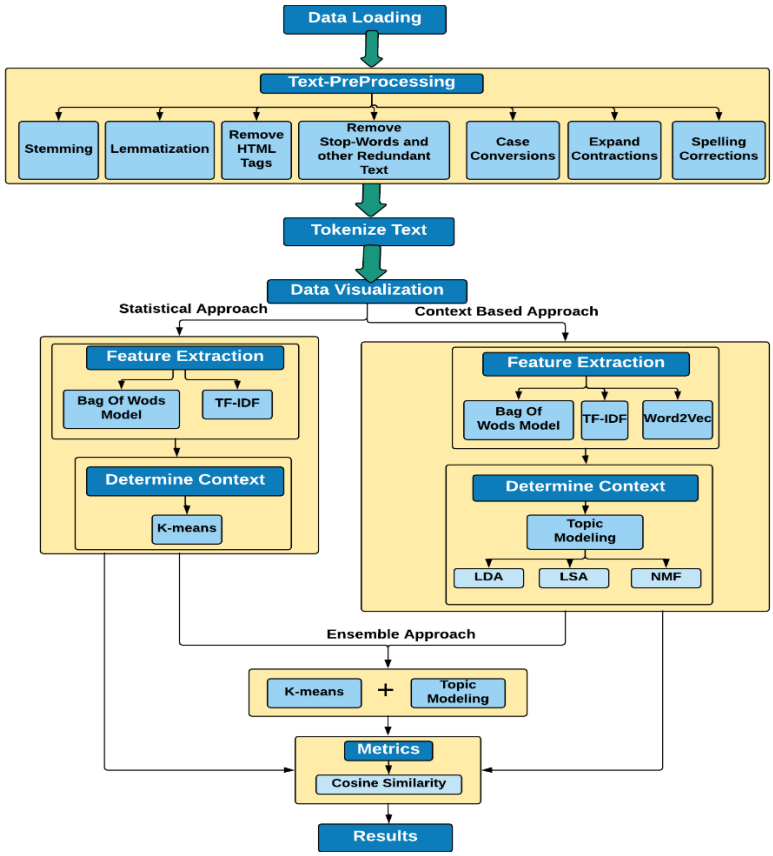

Fig. 4. Model Pipeline representing the complete end-to-end process.

3.4 Feature Extraction

Features are defined as unique, measurable attributes/properties for each observation in the data set. These features are the context/category/tag that accurately and uniquely defines the scope of the question text. The same features when fed into machine learning algorithm detects learning patterns that can be applied on future question text. These features, however, need to be vectorized for the algorithm to optimize and minimize the loss when trying to determine patterns. There are several models to implement the feature extraction techniques such as Word2vec, Bag of Words and TF-IDF. We started with Vector space model, where textual data is represented as numeric vectors of specific terms for the vector dimensions.

Then we tried Bag of words model, which converts the text into a vector that represents the frequency of all the distinct words that are present in the text vector space for that specific text. Hence, the weight of each word is equal to its frequency of occurrence in that text. The model can also be created for n-grams. The problem we found with bag of words is that vectors are based on absolute frequencies of the word occurrences, so if the word occurs a lot across all the question text, it has higher frequency and overshadows the other words that may not occur so frequently. Next, we tried Term frequency and inverse document frequency (TF-IDF), which increases proportionally to the frequency of a word appearing in the text but is offset by the frequency of the word in the corpus. Hence, the weight of a word is adjusted based on how many times it appears in the corpus. One of the other vectorization models for extracting features is Google's word2vec algorithm, which is a neural network-based implementation that learns distributed vector representations of words based on continuous bag of words and skip gram-based architectures. This is much faster than other neural network-based implementations and doesn't need manual labels to create meaningful representation among the words. It uses average TF-IDF weighing to build the vector.

Topic modeling uses more statistical and mathematical modeling approach to extract main concepts from corpus of documents, which in our case was our whole python stack overflow question text. Topic models are probabilistic statistical models that use singular valued decomposition (SVD) and Latent Dirichlet Allocation (LDA), to find latent semantic structures in text that yields topics. The key attribute we conducted feature extraction on is the Question Body.

4. Methodologies

Our study answered user's query by first assigning a relevant label or intent to the question, and then matched user's query with the most similar question in stack overflow corpus. Finally, the top 10 most relevant questions were presented.

The first part of the study used two different approaches: k-means document clustering and topic modeling, to group the questions using the stack-overflow generic python (windows) tags. Clustering utilizes unsupervised clustering algorithms to assign a label, while topic modeling uses probabilistic models to group the topics. Ensemble approach was also used to narrow down the feature space to determine the list of similar questions. In the ensemble approach, documents were clustered using unsupervised k-means and then the trained k-means model was used to conduct topic modeling on top of the clusters. The results were analyzed using both quantitative and qualitative metrics. The second part of the study focused on using different similarity models to find the most similar question. In case of the ensemble approach, the entire corpus of questions was not used to look for similar questions and was limited to the cluster that the user question falls within. The ensemble approach, k-means classifier results and the topic modeling similarity results, as well the retrieved questions were analyzed.

While there are several clustering algorithms, our focus was mainly on unsupervised clustering algorithms, as pre-assigned question tags from stack-overflow were not used to determine our intent. Pre-assigned tags for python-windows were used mainly to filter down to our corpus of interest. Unsupervised learning algorithms draw inferences from datasets without labels.

The most common unsupervised learning methods are clustering and topic modeling. Clustering fragments a collection of documents into clusters, where documents in each cluster are more like each other than those in other clusters. Documents are clustered based on a similarity measure. Topic modeling, on the other hand, creates probabilistic models to determine soft clusters, where every document has a probability distribution over all the clusters. Each topic in topic modeling is represented as a probability distribution over words and each document is expressed as a probability distribution over topics.

4.1 Clustering

Our focus was on partitional clustering. Partitional clustering divides a set of data objects into non-overlapping subsets, where each data object is in exactly one subset. It creates a one-level partitioning of points where the number of clusters "k" is user specified.

The criteria for clustering can differ from one algorithm to another, based on the goal of each algorithm. A well-separated cluster is where a cluster is defined as a set of objects where each object is closer to every other object in the cluster than to any object not in the cluster. There is no specific shape for well separated clusters. In Prototype based clustering, an ideal cluster is where the distance between each object is closer to a specific prototype than to any other cluster. A prototype, in case of continuous attributes, is typically a centroid or average of all the points in the cluster and, in case of categorical attributes, a medoid which is the most representative point of a cluster. Prototype clusters are also referred to as center-based clusters and tend to be more globular in nature.

K-means, which is based on prototype-based clustering, was evaluated. K-means attempts to find a user-specified number of clusters, which are represented by their centroids. The document data is represented as a document-term matrix and the objective is to maximize the cosine similarity of the documents in the cluster to the cluster centroid, also known as maximizing the cohesion of the cluster. First, "k" initial centroids are chosen and then the points are assigned to the initial centroids. Points are assigned to those centroids based on cosine similarity and the centroid is then updated. Again, points are assigned to the updated centroids, recalculated, updated. The algorithm terminates when no more changes occur, and the centroids identify the natural groupings of points. The algorithm used for k-means is formally described as follows.

Input: Document set "D", similarity measure "S", number of clusters "k"
Output: Set of clusters "k"
Initialization: Select random k data points as starting centroids
While not converged ***do:***

…..Assign documents to centroids based on closest similarity
…..Calculate the cluster centroids for all clusters
End
Return "k" clusters

4.2 Determining Clustering Parameters

Determining optimal number of clusters is the key for k-means. While most of the methods are subjective and there is no definitive method, one of the popular methods is the elbow method. k-means clustering defines clusters such that the total intra cluster variation is minimized. Total intra cluster variation measures the compactness of clustering.

Elbow method was used with two different metrics: within cluster sum of square (WCSS) and variance. WCSS measures the total intra cluster variation as a function of the number of clusters. Variance score, which represents the percentage of variance explained by the number of clusters, was also plotted. Typically, when this is plotted, the location of the bend in the plot is an indicator of the optimum number of clusters. Number of clusters was chosen such that the rate of the drop in WCSS levels off and is minimized and the percentage of variance is maximized.

Additionally, gap statistic was computed to determine the ideal number of clusters and cross reference it with the value received from the elbow method. Gap statistic method can be applied to any clustering method. It compares the total within intra-cluster variation of different values of "k" with their expected values under null reference distribution of data. The idea is to maximize the gap statistic, which signifies that clustering structure is far away from the random uniform distribution of points.

After converting the raw, processed documents into term-frequency-inverse document frequency matrix, WCSS was plotted against varying number of clusters for the k- means model. The results are depicted in Fig. 5. It is evident that as the number of clusters increases, WCSS decreases. The rate of drop in WCSS appeared to drop off around k=20 and continued to drop in small increments thereafter.

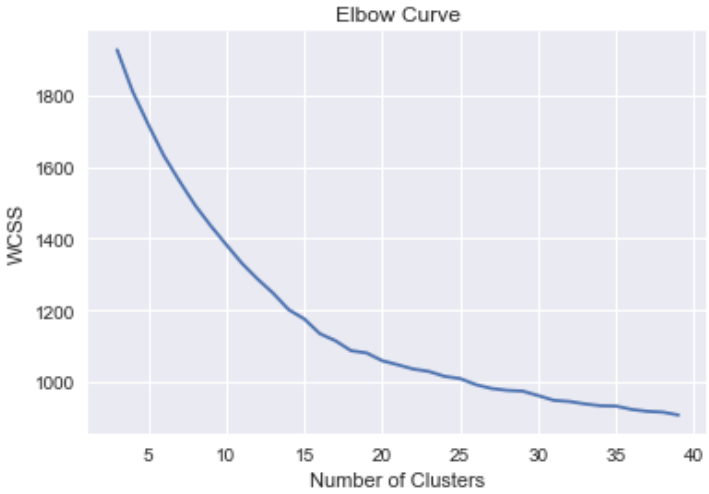

Fig. 5. Elbow Curve for WCSS vs. Number of Clusters

Similarly, per the results from Fig. 6., the variance continued to increase as the number of clusters increased, and an elbow right around k=20 could be seen. Ideally, the number of clusters that maximizes the percentage of variance explained is desired.

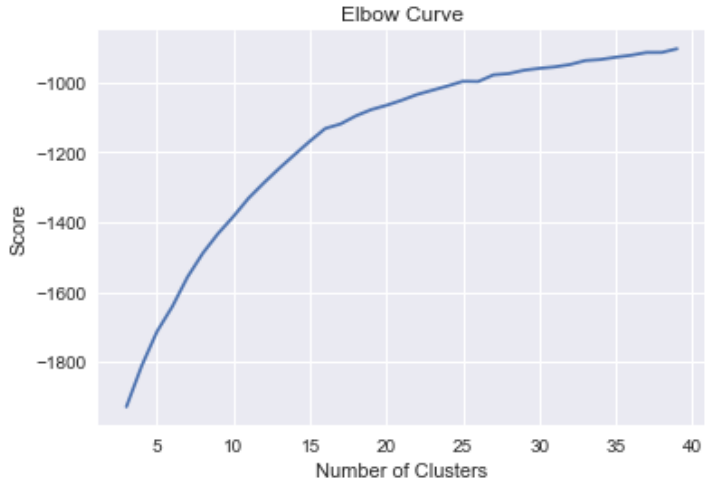

Fig. 6. Elbow Curve for Variance Score vs. Number of Clusters

Next, gap-statistic value was plotted against the number of clusters, as shown in Fig. 7. While based on the plot, gap statistic was maximized at k=40, a similar elbow bend around k=20 can be seen, as with the WCSS and variance plot.

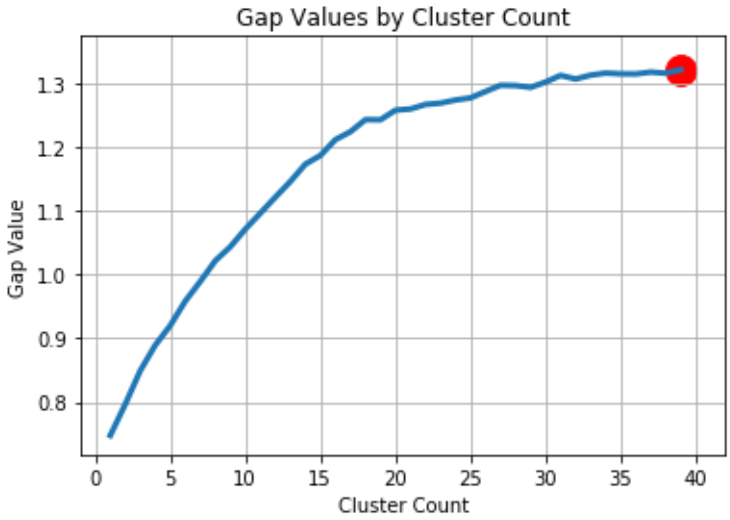

Fig. 7. Elbow Curve for Gap Statistic vs. Cluster Count

Based on the results gathered from the gap statistic, WCSS and variance plot, we continued with k=20 as the optimum number of clusters. While a higher number of clusters may result in better WCSS, variance and gap statistic, too many clusters can result in highly variable clusters and are more difficult to interpret as well.

The distribution of number of items per cluster is displayed in Fig. 8. Apart from cluster 17, most of the other clusters contained items in the range of 100 to 250. Cluster 17 indicated that there were some outliers.

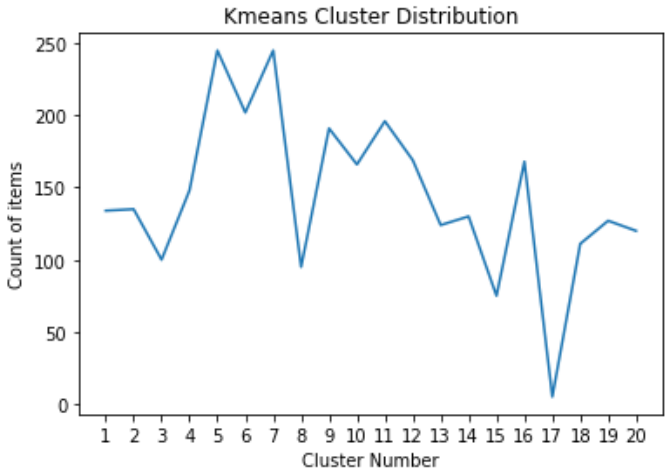

Fig. 8. K-means Cluster Distribution

4.3 Clustering Quantitative Validity

Clustering groups the similar objects within the clusters and keep the objects in different clusters distinct. There are two different types of clustering validations: internal and external. While external validation relies on external data to do the validation, internal validation measures the goodness of the clusters without respect to external information. Our focus is on internal validation. Clustering validation evaluates the goodness of the clusters based on compactness and separation. Compactness measures how closely related objects in a cluster are based on metrics such as variance and distance. Lower variance equates to better compactness. Separation measures how well separated a cluster is from the other clusters. Some common metrics to measure separateness is pairwise minimum distance between objects in different clusters and measures based on density.

Silhouette coefficient and Dunn index were used for our evaluation. Silhouette coefficient measures the performance based on pairwise difference of between and within cluster distances. Number of clusters that maximize the silhouette coefficient is the optimal number. Dunn's index uses the minimum pairwise distance between objects in different clusters as the inter-cluster separation and the maximum diameter among all clusters as the intra-cluster compactness. The optimal number is the number of clusters that maximizes the index value.

Per the results from Table 4., while the Dunn index was highest for lower number of clusters, the silhouette coefficient was very low as well. Allowing for 20 clusters maximized the silhouette coefficient, as well as maintained a high Dunn index. Hence, at k=20, the clusters were most compact, as well as most separated from other clusters.

Table 4. Quantitative Analysis for K-means

Number of Clusters	Silhouette Coefficient	Dunn Index
4	0.12	1.00
10	0.16	1.00
20	0.16	0.93
30	0.15	0.78
40	0.15	0.80
50	0.14	0.80

4.4 Clustering Qualitative Validity

We visualized the 20 clusters and extracted the top 5 highest weighted terms from each cluster to determine the context/label for each cluster. As shown in Fig. 9., there were obvious outliers which appeared to be grouped together. While most of the clusters appeared to be tightly clustered, there were a few that were a little more spread out.

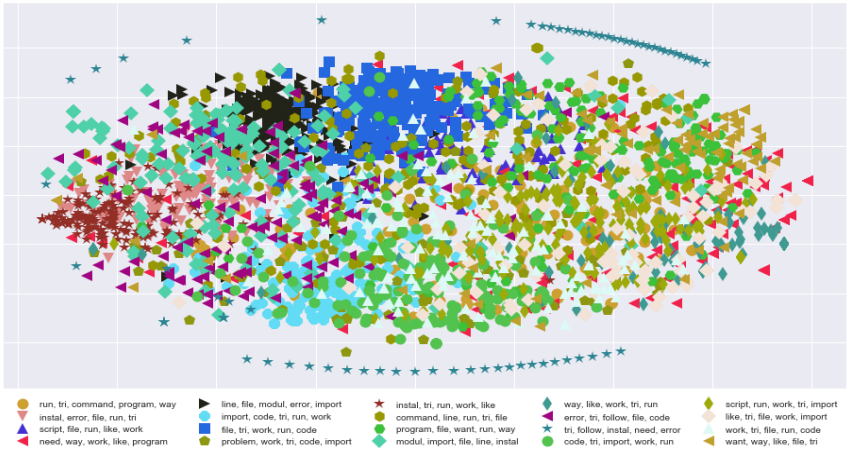

Fig. 9. K-means Clustering

We analyzed each of the 20 clusters in more detail and evaluated the context of documents contained in each cluster. The determined context is highly subjective. The results of our evaluation are shown in Table 5.

Table 5. Cluster Qualitative Evaluation

Cluster	Top 5 terms	Evaluated context
1	run,tri,command, program, way	Running python program and commands
2	install, error, file, run, tri	File installation errors
3	script, file, run, like, work	Running files and batch scripts
4	need, way, work, like, program	General program related questions
5	line, file, modul, error, import	File line reading and module errors
6	import, code, tri, run, work	Code related, Thread related (multi-threading, execution order of threads), execution
7	file, tri, work, run, code	File related questions such as how to open/read files/permissions
8	problem, work, tri, code, import	Problems related to importing and python interfacing with other applications
9	install, tri, run, work, file	"How to" Installation questions
10	command, line, run, tri, file	Commands from command prompt
11	program, file, want, run, way	Programming semantics
12	modul, import, file, line, instal	Code usage and installation error questions
13	way, like, work, tri, run	Monitoring, performance, Configuration
14	error, tri, follow, file, code	Compilation errors
15	tri, follow, instal, need, error	Connectors, Connection questions, Network related
16	code, tri, import, work, run	Operation related such as how to minimize, resize windows, open .bat file from python
17	script, run, work, tri, import	Running Scripts
18	like, tri, file, work, import	Credentials, Webserver and filesystem related
19	work, tri, file, run, code	Python questions such as double clicking not executing, how to enable text highlighting
20	want, way, like, file, tri	Scripting questions

Finally, 10 questions were created and clustered using this K-means model. At this point k-means model was used as a supervised model and clustered the new questions to extract only the meaningful features. Cosine similarity was calculated between the new question and all the questions from the designated cluster. The results are displayed in Table 6.

Cosine similarity calculates the angle between the two documents on the vector space. It measures the orientation, and not the magnitude. Hence, it is a comparison between two documents in a normalized space, as only the angle between the two documents is considered and not the magnitude of each word count of each document. This metric defines how related the two documents are.

As displayed in Table 6. 10 test questions were randomly picked, which might or might not have been in the corpus. Cluster number defines the cluster each question was placed in when we fit the k-means model on the individual questions. Top 10 highest cosine similarities for each of the test questions was retrieved. For Questions 1, 3, 5, and 9, either the exact match or a similar question that may answer the desired question was found, as depicted through the label "Match". However, Question 10 appeared to be incorrectly clustered as the test question was clustered in Cluster 12, but the original was in Cluster 14. There were no matches found for Question 2, 6, 8 and 10. For Question 4 (TensorFlow), a matching question wasn't expected to be found, as that topic was not part of our original trained corpus.

Table 6. Cosine Similarity for Test Questions based on K-means Classifier

Q-No	User Query	Ques matching to User Query using K-means	Cosine Similarity
1	How do I install pip on Windows?	Match	0.4069101
2	Anaconda Installed but Cannot Launch Navigator	No Match	No Match
3	Drag and drop onto Python script in Windows Explorer	Match	0.1729243
4	How can one validate a Tensorflow installation on Windows	No Match	No Match
5	Bug with Python UTF-16 output and Windows line endings?	Match	0.1650546
6	Python process fail after ctypes CreateProcessWithLogonW execution	No Match	No Match
7	Detecting User Idle Time in Python	Match	0.1030193
8	Get a preview JPEG of a PDF on Windows?	No Match	No Match
9	How to capture Python interpreter's and/or CMD.EXE's output from a Python script?	Match	0.5231542
10	What's the best way to duplicate fork () in windows?	No Match	No Match

Even though k-means is guaranteed to converge, it may lead to a local minimum. The convergence and occurrence of local minimum are highly dependent on the initial number of clusters "k". K-means++ scheme has been proven to alleviate this problem. K-means++, as implemented by scikit-learn, initializes the initial centroids to be far apart from each other. We implemented k-means++ and used gap-statistic, variance, and WCSS as the base

metrics for deriving the value "k." While every precaution necessary was taken to derive the true value of "k", there is no guarantee that we got the true cluster representation of the data. Based on our formed clusters, some of the clusters overlapped and hence, some questions were misclassified.

4.5 Topic Modeling

Topic Models are unsupervised algorithms which automatically identifies topics present in text documents and derive hidden patterns in a large corpus of text which assists in better decision making. Topic Modeling is a text mining process where a corpus of unstructured documents becomes the input and the output is the top ranked terms in a topic and the documents associated relative to that topic. In the topic modeling approach, a single text document can be associated with multiple topics compared to document classification where only single topic is associated to a text. Topic modeling builds cluster of words instead of cluster of texts/documents where each text is a mixture of different topics with each topic carrying certain weight to it. In these clusters of words, topics which are closer together are more similar and topics further apart are less similar. When a topic is selected, the most representative words of the selected topic can be seen, and this can be a measure of how frequent or how discriminant the word is. The primary packages used for topic modeling are Gensim, Sci-Kit Learn (Sklearn), Stanford Topic Modeling Toolkit and Mallet but we're focused on Gensim and Sklearn in this study. Gensim scales well to large corpuses, but it does not include Non-Negative Matrix Factorization (NMF) algorithm in Sklearn library, which can also be used to find topics in text documents. The mathematical basis of NMF is quite different from LDA and sometimes found to be performing better than LatentDirichletAllocation (LDA).

LDA is the modeling process that revolves around three things: the text corpus, the collection of documents D and the words W in the documents. LDA is one of the most popular method for doing topic modeling because it provides accurate results, can be trained online, and can run on multiple cores. The LDA algorithm first models documents via a mixture model of topics, then from these topics, words are assigned weights based on the probability distribution of these topics. Because of this probabilistic assignment over words, LDA allows a user to say how likely a word falls into a topic. Subsequently from the collection of words assigned to a specific topic, we are thus able to gain an insight as to what that topic may represent from a lexical point of view.

The generative process is defined [9] as follows:

1) For every topic $k = \{1, ..., K\}$
    a) Draw a distribution over the vocabulary V, $\beta k \sim \text{Dir}(\eta)$
2) For every document d
    a) Draw a distribution over topics, $\theta d \sim \text{Dir}(\alpha)$ (i.e. per-document topic proportion)
    b) For each word w within document d

i) Draw a topic assignment, $zd, n \sim$ Mult(θd), where $zd,n \in \{1, ..., K\}$ (i.e. per-word topic assignment)

ii) draw a word $wd,n \sim$ Mult($\beta zd,n$), where $wd,n \in \{1, ..., V\}$

Each topic βk is a multinomial distribution over the vocabulary V and comes from a Dirichlet distribution $\beta k \sim$ Dir (η). Additionally, every document is represented as a distribution over K topics and come from a Dirichlet distribution $\theta d \sim$ Dir (α). The parameter α denotes the smoothing of topics within documents, and η denotes the smoothing of words within topics. The joint distribution of all the hidden variables βK (topics), θD (per-document topic proportions), zD (word topic assignments), and observed variables wD (words in documents) is expressed by (1):

$$p(\beta_K, \theta_D, z_D, w_D|\alpha, \eta) = \Pi^K{}_{k=1} p(\beta_K|\eta)\, \Pi^D{}_{d=1}\, p(\theta_d|\alpha) \Pi^N{}_{n=1}\, p(z_{d,n}|\theta_d) p(w_{d,n}|z_{d,n}, \beta_{+d,k}) \quad (1)$$

Fig. 10 shows the probabilistic graphical model of LDA in plate notation form [9], where the unshaded nodes in the diagram represent the hidden random variables, the shaded nodes represents the observed random variables, and the edges represents the conditional dependencies between the hidden and observed variables. The rectangles represent replication (plates). The graphical model is equivalent to the collective probability of all the hidden and observed variables expressed in (1).

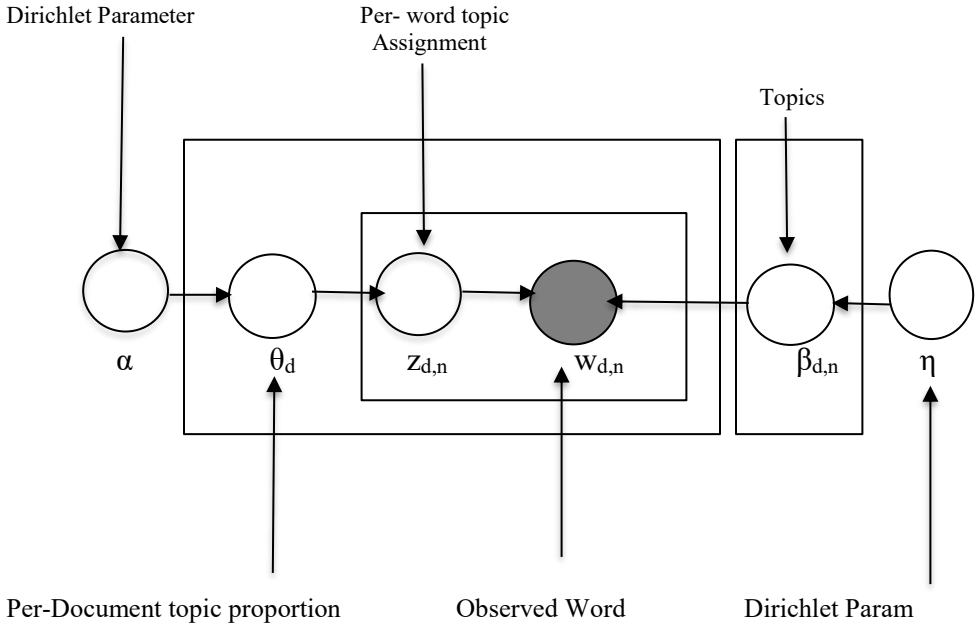

Fig. 10. LDA represented as a graphical model

Non-negative matrix factorization (NMF or NNMF**)** is an algorithm that assumes that the data and its components are non-negative. Non-negative matrix factorization can be used where the data matrix does not contain negative values. In NMF, the representation of a vector is obtained in an additive fashion, by superimposing the components, without subtracting. It finds a decomposition of samples **X** into two matrices **W**and **H** of non-negative elements, by optimizing the distance **d** between **X** and the matrix product **W*H**. The most widely used distance function is the squared Frobenius norm[2], which is an extension of the Euclidean norm to matrices, shown in (2):

$$d_{Fro}\,(X,Y)\;=\;1/2||X-Y||^2{}_{Fro}\;=\;1/2\;\sum_{i,j}\quad(X_{ij}\;-\;Y_{ij})^2 \tag{2}$$

Latent Semantic Analysis or Latent Semantic Indexing (LSA/LSI) makes use of linear algebraic techniques to learn the conceptual correlations in text documents. LSA constructs a weighted term-document matrix which works on a technique called Singular Value Decomposition (SVD) on the matrix which then uses the matrix to identify the concepts contained in the text. When truncated Singular Value Decomposition (SVD) is applied to term-document matrices using either Count Vectorizer or TfidfVectorizer, the transformation is known as latent semantic analysis (LSA), because it transforms term document matrices to a "semantic" space of low dimensionality. Latent Semantic Analysis (LSA) is known to counter the effects of synonymy and polysemy which means that there are multiple meanings of a word causes term-document matrices to be overly sparse and exhibit poor similarity under measures such as cosine similarity. Mathematically, truncated SVD[14] applied to training samples **X** produces a low-rank approximation **X** using (3):

$$X\approx X_k\;=\;U_k\sum_k\quad V^T{}_k \tag{3}$$

After this operation, $U_k\sum_k^T$ is the transformed training set with $\boldsymbol{k}$ features (called n_components in the API). To transform a test set **X**, we multiply it with $\mathbf{V_k}$ using formula shown in Formula 4:

$$X'\;=\;XV_k \tag{4}$$

In this paper, we explored LDA, NMF and LSA algorithms using Gensim and Sklearn libraries. Gensim offers Latent Semantic Analysis and Latent Dirichlet Allocation algorithms which are unsupervised algorithms and only need the corpus of plain texts as the input. LSA transforms documents from either bag-of-words or TfIdf-weighted space into a latent space of a lower dimensionality where TFIDF is a preferred method. LDA is an algorithm where bag-of-words counts are transformed into a topic space of lower dimensionality. LDA is a probabilistic extension of LSA, so topics derived from LDA can be interpreted as probability distributions over words from the collection of text documents. In both LDA and LSA, number of topics need to be provided beforehand. The number of topics were chosen using elbow curve and it was found that 10 was the optimal number of

topics, as can be seen in Fig. 11. After 10 topics some of the topics started pulling the same words and were looking similar.

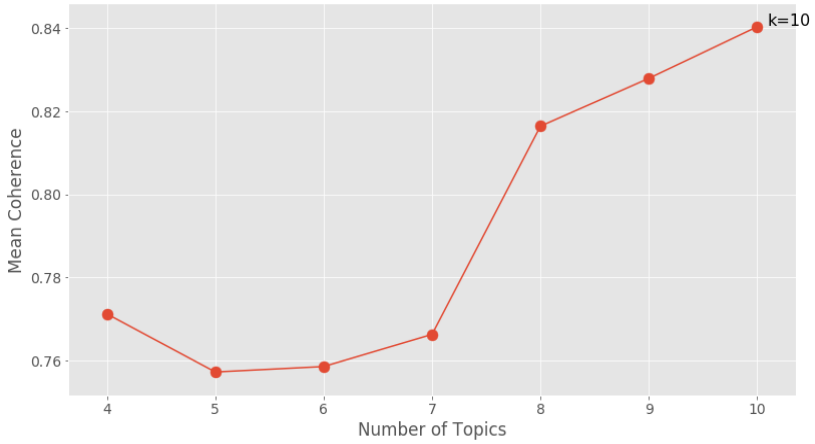

Fig. 11. Elbow curve for finding number of clusters for LDA/LSA

The top 2 topics generated by LDA model are as displayed in Results 1. LDA selected the top 10 words associated with 2 random topics. The floating number attached to each word is the *weight* of the word it is multiplying, i.e., how much each word influenced the topic. The model used id2word dictionary that was provided to the model. It was observed that in the LDA model, "import", "file", "def" script "r" were the most related words and contributed predominantly to the first topic, whereas the second topic is mostly related with "file", "line", "run" and "import". The magnitude of the weights attached to the words can be thought of how important a particular word is in that topic. Or, more loosely, how important the topic is in describing the corpus upon which the model is based.

(0, '0.013*"import" + 0.009*"file" + 0.009*"def" + 0.009*"script" + 0.008*"r" + 0.008*"print" + 0.007*"ode" + 0.007*"like" + 0.006*"get" + 0.006*"run"')
(1, '0.035*"file" + 0.019*"line" + 0.011*"run" + 0.010*"import" + 0.010*"modul" + 0.009*"error" + 0.009*"script" + 0.008*"name" + 0.008*"tri" + 0.008*"ode"')

Results 1. LDA Topics

The top 2 topics generated by LSA/LSI model are as displayed in Results 2. According to LSA/LSI, "C", "tri" , "import", "file" are all related words (and contribute the most to the first topic), whereas the second topic concerns itself primarily with "file", "line", "tri", "run". Furthermore, it was also noticed that LSI selected words with both positive and negative probability for each of the first ten topics. The statistical meaning of a negative value in the topic is that the occurrence of this semantic concept is accompanied by the lack of the word in the underlying corpus. Though it has been quoted by Radim Rehurek

(creator of gensim library) that - “LSI topics are not supposed to make sense; since LSI allows negative numbers, it boils down to delicate cancellations between topics and there's no straightforward way to interpret a topic.”

(0, '0.800*"c" + 0.537*"tri" + 0.175*"import" + 0.096*"file" + 0.095*"precompil" + 0.060*"line" + 0.036*"error" + 0.036*"find" + 0.034*"run" + 0.034*"builtin"'),
(1, '-0.602*"file" + -0.375*"line" + 0.351*"tri" + -0.198*"run" + -0.169*"instal" + -0.158*"find" + -0.156*"error" + -0.115*"modul" + -0.109*"librari" + -0.099*"copi"'),

Results 2. LSA/LSI Topics

Next, we used LDA, LSA and NMF implementations using Sklearn library to derive the topics from the collection of text documents and the snippets for the top-ranked documents for each topic. Again we used elbow curve as a method to find the optimal number of topics for the algorithms. As seen in Fig. 12, 10 topics was a reasonable cut-off point, as that’s where the words within the topics were distinct and interpretable.

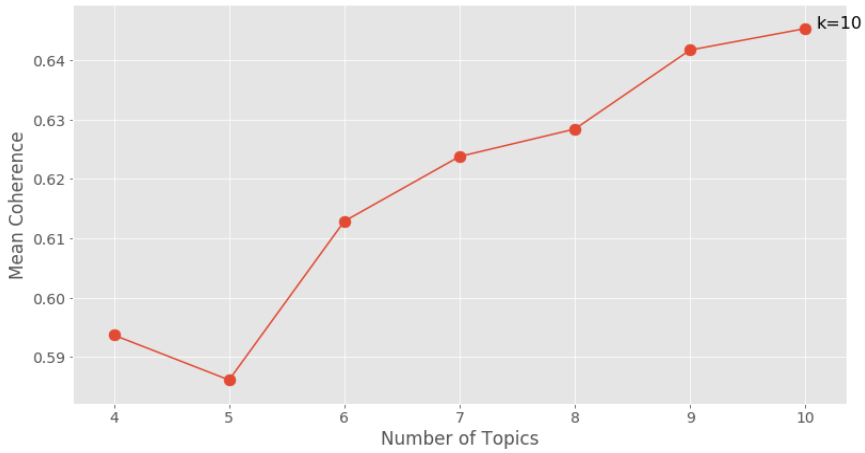

Fig. 12. Elbow curve for finding number of clusters for NMF/LDA/LSA

The top 2 topics generated by LDA are as listed in Results 3.

Topic 01: aes256, debug2, joke, ecdh, tow, hmac, debug1, debug3, cbc
Topic 02: file, instal, run, pi, script, line, import, tri, work, program

Results 3. LDA Topics using Sklearn

The top 2 topics generated by LSA/LSI are as listed in Results 4.

Topic 01: file, instal, pi, run, script, line, program, command, tri, path
Topic 02: lib, instal, pi, line, python27, site, packages, pip, modul, file

Results 4. LSA/LSI Topics using Sklearn

The top 2 topics generated by NMF are as listed in Results 5.

Topic 01: file, open, txt, directori, folder, read, write, creat, copi, filenam
Topic 02: script, run, command, pi, execut, ex, cmd, batch, prompt, consol

Results 5. NMF Topics using Sklearn

We assigned the topic labels by using the term weight frequency for the top 10 words for each topic. All the 3 models using sklearn provided similar weights to the topics assigned. Topic models reflect the structure of the data available. It is recommended that this analysis is used as an exploratory exercise, as it is open to human interpretation.

Once we were done with topic modeling, next task was to measure the success of topic modelling. We conducted a systematic search of the space of coherence measures using Stack Overflow data for the evaluation. Since topic modeling is an *unsupervised* machine learning approach, it is not easy to evaluate the topics in the generated models since there is no labelled "ground truth" data to compare with. Moreover, topic modeling does not give any guarantees on the interpretability of the output. Topic modeling typically requires some parameters to be defined beforehand such as the number of topics k to be discovered. Hence, model evaluation is crucial to find an "optimal" set of parameters for the given data. Model evaluation is hard when using unlabeled data. We used the metric –topic coherence – in this paper to assess a model's quality to find the "best" model. Nevertheless, it is still important to check if the generated model makes sense practically. A good practical method to assess the quality of a model is the human-in-the-loop approach, where a human being must spot manually random "intruder" words or "topic intruders". We chose the good models with a theoretical approach at first using topic coherence. A set of documents/texts is considered coherent, if they support each other.

Topic Coherence is the measure of extent to which the top terms represented in a topic are semantically related to the corpus that is being used as a base. Topic coherence is applied to top n words of each generated topic by the latent variable models. The good models will generate topics which have high coherence scores. To measure coherence, various measures are used, such as c_v, c_uci, c_npmi, u_mass and TC-W2V. In this paper, we used c_v for topic modeling in genism and TC-W2V for sklearn.

4.6 Topic Modeling Quantitative Validity

CV measure in Gensim library is based on four parts i.e. segmentation of the data into pair of words, calculation of word pair probabilities, calculation of a confirmation measure that quantifies how strongly a word set supports another word set, and finally aggregation of individual confirmation measures into an overall coherence score. We set up 5 different LDA topic models- batch with passes=1, passes=50, passes=100 and default passes, and online with default passes. As can be seen in Table 7, topic coherence decreased when we changed the training mode from batch to online. In order to build a "good" topic model, we simply trained it using more passes than the existing model. Therefore, the c_v coherence should in theory be better for the good model than the existing one. Hence as we can see, the c_v coherence for the good LDA model (passes=50) is little better than that for the existing LDA (passes=1) model. This is because the good LDA model usually comes up with better topics that are more human interpretable. The c_v topic coherence capture this wonderfully by giving the interpretability of these topics a number and hence this coherence measure can be used to compare difference topic models based on their human-interpretability.

Table 7. Topic Coherence for Gensim Models

Model	Topic Coherence
LSI	0.409
LDA	0.523
HDP	0.464

Table 8. Topic Coherence for LDA Models

Model	Topic Coherence
online	0.460
Passes=1 (batch)	0.536
Passes=50 (batch)	0.540
Passes=100 (batch)	0.530

TC-W2V measure in Sklearn library uses w2vec model which is neural network-based algorithm for estimating word representations in a vector space using Continuous Bag-of-Words (CBOW), where the current word is predicted based on its context, and Skip-gram, which predicts context words based on the current word. In TC-W2V, the coherence score is calculated as the mean pairwise Cosine similarity of two term vectors generated with a Skip-gram model. As displayed in Table 9, we set up 4 different models here: batch, online, with max_iter=1, max_iter =50 and max_iter =100. LSI, LDA and NMF were improved models using max_iter parameter. Per the results, topic coherence of LDA and NMF models improved with max_iter=50 and remained almost same for iteration=100.

Table 9. Topic Coherence for Sklearn Models

Model	Topic Coherence
LSI	0.619
LDA	0.540
NMF	0.643

Table 10. Comparison of Topic Coherence for LSI/LDA/NMF

Model	Max_iter=1	Max_iter=50	Max_iter=100
LSI	0.540	0.520	0.526
LDA	0.590	0.589	0.589
NMF	0.591	0.669	0.668

Next, similarity between a specific document and a user query was measured. Similarity measures used for this study was cosine similarity. Cosine similarity is a standard measure in Vector Space Modeling, but wherever the vectors represent probability distributions, different distance measures are used like Hellinger distance and Jaccard distance to compare the similarity or diversity of text documents. The cosine similarity metric gives an output in the range [-1, 1] for two sparse vectors, with values closer to 1 which means that they are very similar to each other and closer to -1 means least similar to each other. As seen in Table 11., the same 10 test questions that were used in k-means were used here. In topic modelling, since TFIDF and Count Vectorizer implementations are used which are based on frequency/count of words, the algorithms pull semantically similar documents, so human intervention is required to see which document match makes sense. Per the results, Ques No-1 and 9 were exact match and Ques No- 3 & 5 were semantically similar. Rest of the documents were semantically similar too but after human intervention, it was determined that those document similarities did not make sense, hence, was not counted as a relevant match.

Table 11. Similarity Table of LDA and LSA using Gensim Library

Q-No	User Query	Model	Ques matching to User Query	Cosine Similarity
1	How do I install pip on Windows?	LDA	Match	0.99
		LSA	Match	0.99
2	Anaconda Installed but Cannot Launch Navigator	LDA	No Match	No Match
		LSA	No Match	No Match
3	Drag and drop onto Python script in Windows Explorer	LDA	Match	0.65
		LSA	Match	0.7
4	How can one validate a Tensorflow installation on Windows	LDA	No Match	No Match
		LSA	No Match	No Match
5	Bug with Python UTF-16 output and Windows line endings?	LDA	Match	0.89
		LSA	Match	0.87
6	Python process fail after ctypes CreateProcessWithLogonW execution	LDA	No Match	No Match
		LSA	No Match	No Match
7	Detecting User Idle Time in Python	LDA	No Match	No Match
		LSA	No Match	No Match
8	Get a preview JPEG of a PDF on Windows?	LDA	No Match	No Match
		LSA	No Match	No Match
9	How to capture Python interpreter's and/or CMD.EXE's output from a Python script?	LDA	Match	0.99
		LSA	Match	0.89
10	What's the best way to duplicate fork () in windows?	LDA	No Match	No Match
		LSA	No Match	No Match

Next, we performed similarity measures using Sklearn. Euclidean distance was used for finding the distance between query document and the documents pulled by the indexed documents. Cosine similarity was used to measure the similarity between the query document and the indexed document. As can be seen in the Table 12. below, that Ques No-1 was the exact match and Ques No- 3, 5 and 9 were semantically similar. Similar to Gensim, rest of the questions didn't have a relevant match.

Table 12. Similarity Table of NMF, LDA and LSA using Sklearn library

Q-No	User Query	Model	Ques matching to User Query	Cosine Similarity
1	How do I install pip on Windows?	NMF	Match	0.99
		LDA	Match	0.99
		LSA	Match	0.99
2	Anaconda Installed but Cannot Launch Navigator	NMF	No Match	No Match
		LDA	No Match	No Match
		LSA	No Match	No Match
3	Drag and drop onto Python script in Windows Explorer	NMF	Match	0.7
		LDA	Match	0.65
		LSA	Match	0.7
4	How can one validate a Tensorflow installation on Windows	NMF	No Match	No Match
		LDA	No Match	No Match
		LSA	No Match	No Match
5	Bug with Python UTF-16 output and Windows line endings?	NMF	Match	0.8
		LDA	Match	0.89
		LSA	Match	0.87
6	Python process fail after ctypes CreateProcessWithLogonW execution	NMF	No Match	No Match
		LDA	No Match	No Match
		LSA	No Match	No Match
7	Detecting User Idle Time in Python	NMF	Match	0.8
		LDA	Match	0.85
		LSA	Match	0.75
8	Get a preview JPEG of a PDF on Windows?	NMF	Match	0.8
		LDA	Match	0.7
		LSA	Match	0.8
9	How to capture Python interpreter's and/or CMD.EXE's output from a Python script?	NMF	Match	0.8
		LDA	Match	0.89
		LSA	Match	0.87
10	What's the best way to duplicate fork () in windows?	NMF	No Match	No Match
		LDA	No Match	No Match
		LSA	No Match	No Match

4.7 Ensemble

Ensemble is a technique where two or more algorithms of similar or dissimilar types are combined. Ensemble methods usually produce more accurate results than a single model would. Ensemble modeling is not a "modeling" approach in the same sense as regression or logical regression modeling. Instead, it is simply the practice of using multiple models or modeling techniques. Ensemble modeling is commonly used in scenarios where a single modeling technique may be inadequate to account for the wide range of factors that influence an outcome. Whatever way we use to combine different models, it has been generally observed that ensemble models give more accurate results than individual models. This has been the new normal in the predictive modeling world.

Document clustering and topic modelling are closely related tasks that can be used together and can benefit from each other. In this study, we combined document clustering with k-means and topic modelling algorithms using genism to build an ensemble model as proposed in this[1] paper. For a given corpus we assume that there are several groups and each document belongs to one of these groups. Each of the groups consist of a set of local topics that capture the semantics of the given document. We basically identified the cluster for a new document and only fed the corpus belonging to that cluster to the topic modelling algorithm and identified a similar document.

4.8 Ensemble Quantitative Validity

Table 13. Similarity Table of Ensemble Model

Q-No	User Query	Ques matching to User Query using LSA	Ques matching to User Query using LDA	Cosine Similarity LDA	Cosine Similarity LSA
1	How do I install pip on Windows?	Match	Match	0.91	0.91
2	Anaconda Installed but Cannot Launch Navigator	Match	Match	0.94	0.94
3	Drag and drop onto Python script in Windows Explorer	Match	No Match	0.92	NA
4	How can one validate a Tensor Flow installation on Windows	Match	Match	.95	0.95
5	Bug with Python UTF-16 output and Windows line endings?	Match	No Match	0.85	NA
6	Python process fail after ctypes CreateProcessWithLogonW execution	No Match	No Match	NA	NA
7	Detecting User Idle Time in Python	No Match	No Match	NA	NA
8	Get a preview JPEG of a PDF on Windows?	No Match	No Match	NA	NA
9	How to capture Python interpreter's and/or CMD.EXE's output from a Python script?	Match	No Match	0.76	NA
10	What's the best way to duplicate fork () in windows?	No Match	No Match	NA	NA

From the results as seen in Table 13, we noticed that the ensemble only matched exactly for two of the questions and a total similarity match of 6 questions was observed using the genism library. Overall, the LSI model seemed to perform better overall using the ensemble model defined above.

5. Evaluation

In this study, we compared multiple similarity vectorization models that are used to find similar questions using topic modeling techniques, and then produced the most relevant questions that are successfully retrieved. We measure the relevancy for the similarity of questions using recall. We used the cosine similarity for k-means, topic modeling and ensemble models to retrieve the top 10 most relevant questions for each user query. The formula used to calculate recall (4). Relevant document is defined as the total number of questions that are there in the corpus. Retrieved document is defined as question that matched with the relevant document. In the study, there are 9 relevant documents as question number 4 is not in the corpus. The k-means model generated recall of 50%, while topic modeling and ensemble models generated the same recall of 67%.

$$\boldsymbol{Recall} = \frac{|\{\boldsymbol{relevant\ documents}\} \cap \{\boldsymbol{retreived\ documents}\}|}{|\{\boldsymbol{relevant\ documents}\}|} \quad (4)$$

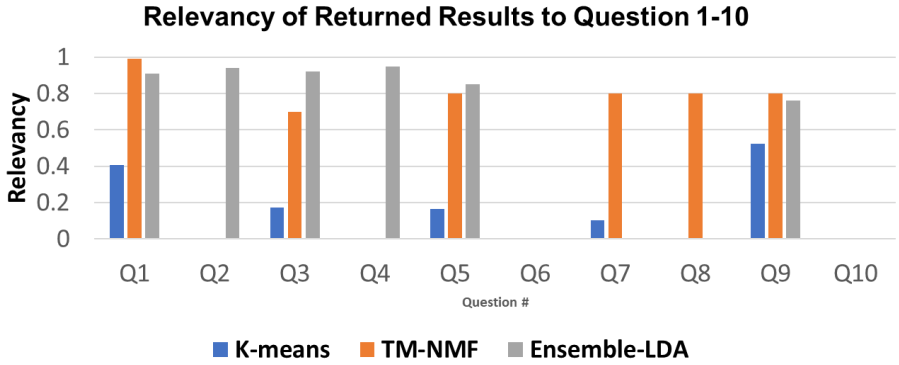

Fig. 13. Cosine Similarity(Relevancy) comparison of results for test questions

In addition, as per Fig. 13, based on the comparison of cosine similarities for k-means, topic modeling (NMF), and Ensemble, topic modeling (NMF) and Ensemble produced the best results. There is higher semantic similarity found with topic modeling and ensemble. While none of the models performed well for question 6 and 10, ensemble alone performed the best for question 2 and 6.

5. Ethics

Stack Overflow is an online question-and-answer website for technical users where users can ask and reply to questions posted by other users, tag and edit questions in collaboration, vote on the quality of questions and answers, and post comments on individual questions and answers. The unprecedented growth of this global question answering platform has brought new challenges of ethics and trust. Stack Overflow repository data is available under Creative Commons Attribution Share-Alike and is publicly available. However, if the user posting the question/answer uses any company related code or exposes any personal information about themselves, it can lead to breach of privacy and exposure of confidential information.

Per the publication authored by Ahmed and Srivastava[4] in 2017, the person's spatial characteristics, race, gender, and status effects the level of response the user receives from the forum. This makes the Stack Overflow platform heavily biased and unethical in terms of neutrality of service. Since Stack Overflow is an open forum, it is possible that users use this platform for marketing or provide inaccurate answers on purpose. Furthermore, due to the competition, if one is aiming to get good score, then to answer first, the quality of the answer is not always good. Getting good score also results in getting good reputation points, which results in cases of collusion and use of unfair means to gain reputation points.

The assumptions of an open question answer platform are that the programmers are thorough, professional, neutral, and unbiased when they are answering questions. However, when these assumptions are violated and abused, it can result in an unethical usage of the platform.

6. Conclusion

The optimization of k-means clustering is highly dependent on initialization of number of clusters "k." If "k" is specified incorrectly, it leads to a local minimum and may not give a true representation of the clusters in the data. Based on the similarity results from k-means clustering, some of the questions were misclassified and some of the clusters had overlap of information. Thus, we explored topic modeling to build more context-based topics. Since topic modeling provides topics present in each text document, topic modeling generates a representation for the set of text documents in the topic space. Topic modeling using Sklearn library produced more coherent topic descriptors than those generated by the Gensim library with higher levels of topic generality and redundancy observed with the latter. Our results indicate that Sklearn library is more suitable for our dataset since Sklearn performed better and gave us better accuracy than Gensim. Document clustering and topic modeling are highly correlated. We coupled the k-means model with topic modeling to create an ensemble implementation. K-means produced a recall of 50% while Topic modeling using NMF algorithm from Sklearn, and the Ensemble implementation both produced a model with a recall of 67%. Furthermore, comparison of cosine similarity

produced similar results with topic modeling and ensemble outperforming k-means. Data Science approaches should be used to identify appropriate results in open question-answer forum and combination of multiple approaches should be used to produce a more accurate model. These results are based on a corpus of 5000 documents. However, since, this is an unsupervised classification, a larger corpus may lead to higher accuracy. This is a possible direction for future research.

7. Future Work

The next steps would be to use the entire corpus to find similar questions and extract the correctly marked answers. Python is a high-level mature programming language for general-purpose programming with a large and active community of developers. Python has a design philosophy that emphasizes code readability and syntax constructs that allow programmers to express logic in fewer lines of code than might be possible with other similar languages like C++ or Java. However, since python is an interpreted language, it is considered too slow for high performance tasks. This is where CUDA comes into play. CUDA is a toolkit that acts as an extension of C with its own programming model that lets programmers run their code on NVDIA's GPU. The key part of CUDA code is the kernel program. The kernel is a function that can be executed in parallel on the GPU. It is executed by CUDA threads and all of them run the same code in parallel. Lastly, we used the Spark standalone mode, which is Spark's built-in clustered environment and is the simplest method to run Spark applications in clustered environment. Table 17. displays the results of the metric comparison between Pandas, CUDA and PySpark conducted on the Stack Overflow Python (Django subcategory) dataset.

Table 17. Metric Comparison

Method	# Of Records	Outcome
Pandas – Binary Vectorization	607K	Memory Error
Pandas (Django subcategory)	55K	Memory Error
Pandas (Django subcategory)	50K	3 minutes
Cuda	No NLTK libraries (N/A)	
PySpark	50K	In Process

Furthermore, discovered tags can be used as intents and then those tags can be used to build a chatbot. Stack Overflow is built as a question/answer engine, which is impersonal and can be cumbersome, depending on how many users have answered a question. The way the webpage is built, if the user has marked the answer as the correct answer, it is not

displayed at the top. Often, the user may have to scroll down through all the answers before finding the correctly marked answer. Also, the multiple answers provided may not even be relevant to user's query, in which case, the click rate to reach the correct answer could be very large. Chat bots provide fast support and provide a balanced approach between providing relevant automated answers and a personable medium. This can result in increased productivity, increased onboarding and increased customer satisfaction.

References

1. Sarkar, Dipanjan: Text Analytics With Python – A Practical Real-World approach to gaining, actionable insights from your data (2016)
2. T. D. Hien, D. V. Tuan, and P. V. At, "Additive update algorithm for non-negative matrix factorization," Intl. Journal of Computer Science and Information Security, vol. 10, no. 9, pp. 1-7, Sep. 2012.
3. W. L. Buntine, "Operations for Learning with Graphical Models," Journal of Artificial Intelligence Research, vol. 2, pp. 159–225, nov 1994.
4. Ahmed, T. & Srivastava, A. Hum. Cent. Comput. Inf. Sci. (2017) 7: 8
5. C. Chen, Z. Xing, X. Wang, "Unsupervised software-specific morphological forms inference from informal discussions", Proc. Int' l. Conf Software Eng., pp. 450-461, 2017
6. Fatemeh Riahi, Zainab Zolaktaf, Mahdi Shafiei, Evangelos Milios, Finding expert users in community question answering, Proceedings of the 21st International Conference on World Wide Web, April 16-20, 2012, Lyon, France
7. C. Stanley and M. D. Byrne. Predicting Tags for StackOverflow Posts. In Proceedings of ICCM 2013 (12th International Conference on Cognitive Modeling), 2013
8. R. Řehůřek, "Scalability of Semantic Analysis in Natural Language Processing," phdthesis, Masaryk University, 2011
9. S. Syed and M. Spruit, "Full-Text or Abstract? Examining Topic Coherence Scores Using Latent Dirichlet Allocation," 2017 IEEE International Conference on Data Science and Advanced Analytics (DSAA), Tokyo, 2017, pp. 165-174
10. Michael Röder, Andreas Both, Alexander Hinneburg, Exploring the Space of Topic Coherence Measures, Proceedings of the Eighth ACM International Conference on Web Search and Data Mining, February 02-06, 2015, Shanghai, China
11. Derek O'Callaghan, Derek Greene, Joe Carthy, Pádraig Cunningham, An analysis of the coherence of descriptors in topic modeling, Expert Systems with Applications: An International Journal, v.42 n.13, p.5645-5657, August 2015
12. Sidorova, Anna & Evangelopoulos, Nicholas & Valacich, Joseph & Ramakrishnan, Thiagarajan. (2008). Uncovering the Intellectual Core of the Information Systems Discipline. MIS Quarterly. 32. 467-482. 10.2307/25148852
13. Pengtao Xie and Eric P Xing. 2013. Integrating document clustering and topic modeling. Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence
14. Bosner, N.: Fast Methods for Large Scale Singular Value Decomposition. PhD thesis, Department of Mathematics, University of Zagreb (2006)